PR #37646 完整报告

vllm-project/vllm

[ROCm][FEAT] AITER Fused Allreduce + RMSNorm

合并时间：2026-05-01 23:07

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/37646

执行摘要

- 一句话：在 ROCm 平台通过 AITER 融合 all-reduce 与 RMSNorm，提升推理吞吐
- 推荐动作：值得精读。该 PR 展示了如何通过 torch.compile 的 pattern matcher 将外部自定义内核无缝集成到 vLLM 的编译图中，同时优雅地处理与 FlashInfer 后端的共存和 CUDA 图捕获。ROCm 团队应关注 compile range 的合理性以及在高并发下的性能表现。关注点设计（is_applicable_for_range 与 compile range 的配合）对其他融合 pass 的引入有参考价值。

功能与动机

整合 AITER 的 fused all-reduce + RMSNorm 内核到 vLLM，减少 all-reduce 与 RMSNorm 之间的核启动和数据传输开销，在 ROCm 平台上获得端到端性能提升。PR 作者在 body 中说明 'This PR integrates AITER fused all-reduce + RMSNorm kernel for ROCm platforms'，并通过 benchmark 展示了在中低并发下的明显加速。

实现拆解

1. 在 vllm/_aiter_ops.py 中定义了 AiterCustomAllreduceProto 协议（声明 fused_ar_rms 等方法），并注册了自定义 op rocm_aiter_fused_allreduce_rmsnorm，包含真实实现 _rocm_aiter_fused_allreduce_rmsnorm_impl 和 fake 实现用于编译。
2. 在 vllm/compilation/passes/fusion/allreduce_rms_fusion.py 中新增了两个模式匹配类：AiterAllreduceFusedRMSNormPattern（匹配 allreduce + rms_norm）和 AiterAllreduceFusedAddRMSNormPattern（匹配 allreduce + fused_add_rms_norm），并实现了 RocmAiterAllReduceFusionPass，它继承自 AllReduceFusionPass，使用 AITER 的 fused 操作替换子图。
3. 在 vllm/compilation/passes/pass_manager.py 中根据 rocm_aiter_ops.is_enabled() 条件来选择 RocmAiterAllReduceFusionPass 而非 AllReduceFusionPass。
4. 在 vllm/config/vllm.py 中扩展了 enable_allreduce_rms_fusion：在 ROCm 平台上自动启用（若 AITER 支持且 TP > 1），并调整了 _set_compile_ranges 以使用 AITER 的 max_size 计算 compile range 端点。
5. 在 vllm/distributed/parallel_state.py 的 graph_capture 中增加了 AITER CustomAllreduce.capture() 的上下文管理器，使得 CUDA 图捕获能正确录制 fused 内核。

6. 测试方面：在 tests/compile/passes/distributed/test_fusion_all_reduce.py 中扩展了测试用例，支持 AITER 融合路径并跳过 ROCm 上不支持的量化变体；在 tests/compile/fusions_e2e/test_tp2_ar_rms.py 中在 CUDA 和 ROCm 上运行端到端测试。CI 配置 .buildkite/test-amd.yaml 添加了对应测试项。

关键文件：

- vllm/compilation/passes/fusion/allreduce_rms_fusion.py（模块 编译优化；类别 source；类型 core-logic；符号 AiterAllreduceFusedRMSNormPattern, init, get_inputs, pattern）：核心实现文件，新增 AITER 专用的模式匹配类（AiterAllreduceFusedRMSNormPattern、AiterAllreduceFusedAddRMSNormPattern）和 RocmAiterAllReduceFusionPass，负责在 torch.compile IR 中识别并替换融合子图。
- vllm/_aiter_ops.py（模块 AITER 接口；类别 source；类型 dependency-wiring；符号 AiterCustomAllreduceProto, capture, close, fused_ar_rms）：定义了 AITER CustomAllreduce 的协议接口，并实现了 fused allreduce + rmsnorm 的 real 和 fake 算子，是融合内核的底层桥梁。
- vllm/config/vllm.py（模块 配置；类别 source；类型 dependency-wiring）：在 enable_allreduce_rms_fusion 中针对 ROCm 添加自动启用逻辑，并在 _set_compile_ranges 中处理 AITER 的 max_size 以拆分 compile 范围。
- vllm/distributed/parallel_state.py（模块 分布式；类别 source；类型 dependency-wiring）：在 graph_capture 中添加 AITER capture context，确保 CUDA 图录制包含 fused 内核。
- vllm/compilation/passes/pass_manager.py（模块 编译优化；类别 source；类型 dependency-wiring）：根据 rocm_aiter_ops.is_enabled() 选择使用 RocmAiterAllReduceFusionPass 还是原有的 AllReduceFusionPass。
- tests/compile/passes/distributed/test_fusion_all_reduce.py（模块 测试；类别 test；类型 test-coverage）：扩展测试用例覆盖 AITER 融合路径，跳过 ROCm 不支持的 FP8/FP4 量化变体。
- tests/compile/fusions_e2e/test_tp2_ar_rms.py（模块 测试；类别 test；类型 test-coverage）：端到端测试在 ROCm 上使用 ROCM_ATTN 和 ROCM_AITER_UNIFIED_ATTN 后端执行融合验证。
- .buildkite/test-amd.yaml（模块 CI；类别 config；类型 configuration）：在 AMD CI 中加入新的端到端融合测试项。

关键符号：AiterAllreduceFusedRMSNormPattern, AiterAllreduceFusedAddRMSNormPattern, RocmAiterAllReduceFusionPass, _rocm_aiter_fused_allreduce_rmsnorm_impl, initialize_aiter_allreduce, enable_allreduce_rms_fusion, graph_capture

关键源码片段

vllm/compilation/passes/fusion/allreduce_rms_fusion.py

核心实现文件，新增 AITER 专用的模式匹配类（AiterAllreduceFusedRMSNormPattern、AiterAllreduceFusedAddRMSNormPattern）和 RocmAiterAllReduceFusionPass，负责在 torch.compile IR 中识别并替换融合子图。

```
# 路径：vllm/compilation/passes/fusion/allreduce_rms_fusion.py

class AiterAllreduceFusedRMSNormPattern(BasePattern, VllmPatternReplacement):
    """匹配 allreduce + rms_norm 子图并替换为 AITER fused 操作。"""
    def __init__(
        self,
        epsilon: float,
        dtype: torch.dtype,
        device: str | None,
        use_aiter_rmsnorm: bool = True,
    ) -> None:
        super().__init__(dtype, device)
        self.dtype = dtype
        self.epsilon = epsilon
        # 从 rocm_aiter_ops 获取注册的 fused allreduce + rmsnorm 自定义 op
        self.FUSED_AR_RMSNORM_OP = rocm_aiter_ops.get_fused_allreduce_rmsnorm_op()

    def get_inputs(self) -> list[torch.Tensor]:
        # pattern 匹配需要示例输入形状，这里使用 (5, 16) 作为 input 和 (16,) 作为 weight
        return [self.empty(5, 16), self.empty(16)]

    @property
    def pattern(self):
        def _pattern(
            input: torch.Tensor, weight: torch.Tensor
        ) -> tuple[torch.Tensor, torch.Tensor]:
            allreduce_output = tensor_model_parallel_all_reduce(input)
            rms = vllm.ir.ops.rms_norm(allreduce_output, weight, self.epsilon)
            return rms, allreduce_output
        return _pattern

    @property
    def replacement(self):
        def _replacement(
            input: torch.Tensor, weight: torch.Tensor
        ) -> tuple[torch.Tensor, torch.Tensor]:
            residual = torch.empty_like(input)
            allreduce = self.FUSED_AR_RMSNORM_OP(
                input_=input,
                residual=residual,
                weight=weight,
                epsilon=self.epsilon,
            )
            return allreduce[0], allreduce[1]
        return _replacement
```

vllm/_aiter_ops.py

定义了 AITER CustomAllreduce 的协议接口，并实现了 fused allreduce + rmsnorm 的 real 和 fake 算子，是融合内核的底层桥梁。

```
# 路径：vllm/_aiter_ops.py

class AiterCustomAllreduceProto(Protocol):
    """AITER CustomAllreduce 的接口协议，用于类型标注。"""
    max_size: int
    world_size: int
    fully_connected: bool

    @contextmanager
    def capture(self): ...
    def close(self) -> None: ...
    def fused_ar_rms(
        self,
        inp: torch.Tensor,
        res_inp: torch.Tensor,
        *,
        w: torch.Tensor,
        eps: float,
        registered: bool = False,
        use_1stage: bool = False,
    ) -> tuple[torch.Tensor, torch.Tensor]: ...
    def should_custom_ar(self, inp: torch.Tensor) -> bool: ...



def _rocm_aiter_fused_allreduce_rmsnorm_impl(
    input_: torch.Tensor,
    residual: torch.Tensor,
    weight: torch.Tensor,
    epsilon: float,
) -> tuple[torch.Tensor, torch.Tensor]:
    aiter_ar = rocm_aiter_ops.get_aiter_allreduce()
    assert aiter_ar is not None, "aiter allreduce must be initialized"

    total_bytes = input_.numel() * input_.element_size()
    hidden_dim = input_.shape[-1]
    token_num = input_.shape[0]
    # AITER 内核支持的 hidden dim 集合（与 one-stage 一致）
    hidden_ok = hidden_dim in (512, 1024, 2048, 4096, 7168)
    # 单次调用 token 数上限（超出时 fallback 到 non-fused）
    token_ok = token_num <= 80
    world_size = aiter_ar.world_size
    full_nvlink = aiter_ar.fully_connected

    # 根据 world size 和 nvlink 决定是否满足 size 要求
    if world_size == 2:
        size_ok = True
```

```
    elif full_nvlink and world_size <= 4:
        size_ok = total_bytes < 256 * 1024
    elif full_nvlink and world_size <= 8:
        size_ok = total_bytes < 128 * 1024
    else:
        size_ok = False

    use_1stage = hidden_ok and token_ok and size_ok

    result = aiter_ar.fused_ar_rms(
        input_,
        residual,
        w=weight,
        eps=epsilon,
        registered=torch.cuda.is_current_stream_capturing(),
        use_1stage=use_1stage,
    )
    assert result is not None
    return result[0], result[1]
```

评论区精华

review 中主要讨论了以下几点：

- token 阈值与性能退化：ProExpertProg 建议直接移除 is_applicable_for_shape 对 token 数的硬限制，但 vllmellm 基于 benchmark 测试（高并发时融合反而慢 10%）保留了安全阈值，并以 max_token_num 注入 compile ranges。
- compile ranges 拆分：ProExpertProg 明确要求将 max_token_num 加入 compile range 端点，避免超出范围时仍进行模式匹配。最终通过 _set_compile_ranges 实现。
- 测试条件反转：gemini-code-assist 指出测试文件中选择 pass 的逻辑 AllReduceFusionPass(vllm_config) if use_aiter else RocmAiterAllReduceFusionPass 是反的，应修正为 RocmAiterAllReduceFusionPass(vllm_config) if use_aiter else AllReduceFusionPass。该问题已在后续提交中修复。
- AITER 支持范围：attila-dusnoki-htec 指出该 PR 未覆盖 DeepSeek 模型（hidden_dim 7168 和 token 数限制）。rbrugaro-amd 后续补充了 hidden_dim 7168 的 one-stage 支持并调整了阈值。
- 设计统一：ProExpertProg 建议将 AITER fusion pass 与 FlashInfer fusion pass 共享基类和工具函数，但最终因差异较大未完全合并，仅继承了 AllReduceFusionPass 的接口。
  - 融合 pass 的 token 阈值与 compile ranges 拆分 (performance): 保留阈值，通过 compute_compile_ranges 添加 max_token_num 端点，实现拆分。
  - 测试中选择 fusion pass 的逻辑反转 (correctness): 已修正方向。
  - AITER 融合 pass 不适用于 DeepSeek R1 模型 (design): rbrugaro-amd 补充了 hidden_dim 7168 的 one-stage 支持，整合进同一分支。
  - 在 graph_capture 中加入 AITER capture context (design): 采用简单 if check，通过了 code review。

- 通过 vllm bench latency 确定 token 阈值 (performance): vllmellm 提供了详细 benchmark，显示 1-256 batch 大部分有加速，仅在 256+ 有轻微退化，最终保留阈值但设置为较大的 max_num_token（基于 max_size / hidden_size 计算）。

风险与影响

- 风险：
  1. 性能退化边界：在高并发（batch > 128）时，融合操作在 benchmark 中有 -10% 左右的退化，目前已通过 max_token_num compile range 限制在大 token 数时回退，但边界值需实测调整。
  2. AITER 版本依赖：强依赖 aiter >= 0.1.10.post3 的 fused_ar_rms API，版本迁移可能破坏兼容。
  3. ROCm only：该特性仅适用于 AITER 支持的 AMD GPU（MI300+），CUDA 用户无影响。
  4. CUDA 图捕获：在 graph_capture 中新增 AITER capture context，若 AITER 版本未正确实现 capture 方法可能会导致图录制失败。
  5. 符号推断精度：_rocm_aiter_fused_allreduce_rmsnorm_fake 的实现简单返回空张量，若未来追加对输出值的检查可能触发问题。 - 影响：仅影响 ROCm 平台且满足 VLLM_ROCM_USE_AITER=1、tensor_parallel_size > 1 的用户。在该场景下，选定的模块（如 Qwen3-30B）在中低并发（8-64 批大每秒 token 吞吐提升 1.5%-13.1%，TTFT 降低 2%-6%）。对 CUDA 和 NCCL 路径完全透明。团队需维护新增加的 AITER 分支和测试，但整体改动集中且无向后兼容问题。 - 风险标记：性能退化边界，ROCm 平台专属，AITER 版本依赖，CUDA 图捕获兼容性

关联脉络

- 暂无明显关联 PR